useImperativeHandle:

Connect with Me:
LinkedIn : https://www.linkedin.com/in/priya-bagde
GitHub : https://github.com/priya42bagde
LeetCode : https://leetcode.com/priya42bagde/
YouTube Channel : https://youtube.com/channel/UCK1_Op30_pZ1zBs9l3HNyBw (Priya Frontend Vlogz)

- Data is passed **from parent to child components via props**, known as unidirectional data flow. **In this case, the parent component cannot access or call a function, or value from a child component directly** and sometime we want the parent component to reach down to the child component such as for **form validation or handling of user input**, to **get the data that originates in the child component for the reusability.** We can achieve through useImperativeHandle hook, which allows us **to expose a value, state, or function inside a child component to the parent component through ref**, which is giving the parent component **more control over** the child component and an **explicit way**.
- You can use when you need to **access a custom component's internals in a way that is not possible using props**.
- Remember that **the ref object should be created in the parent component and passed to the child component using React.forwardRef.** useImperativeHandle must be **wrapped inside the forwardRef.**

Syntax: useImperativeHandle(ref, createHandle, [dependencies])

we can directly use useImperativeHandle **without constant**. We will use this hook **in the child class** because here **ref is passed down from the parent component and it should be created in the parent component.**, createHandle is a **variable or function name, or we can say data which we want to access in the parent component**. dependencies are the **array of dependencies that will occur to re-render. useImperativeHandle does not return anything.**

Use cases:

- When you need a **bidirectional data and logic flow**, but you don't want to overcomplicate things by introducing state management libraries.
- You might programmatically **focus the first input field of a form as soon as a user lands on it, instead of having the user click into the form to focus** it.

React.forwardRef

- **React.forwardRef** allows you **to pass a ref from a parent component to a child component**, enabling you to **access and interact with the child component's DOM elements or instance methods**. This is particularly useful when you need **to control or manipulate elements or behaviors inside a child component directly from its parent component.**
- It allows you to create **higher-order components (HOCs) or wrapper components**, which **can accept a ref and pass it down to an underlying child component**.

- **useRef vs useImperativeHandle:** When compared to **useRef**, **useImperativeHandle** allows for more direct interaction with child components. You would use useRef if you only need **to access the state of the child component**, and you would use useImperativeHandle if you need **to interact with the child component in a more direct way.**

Aspect	`useImperativeHandle`	`useRef`
Purpose	Exposes specific methods or properties of a child component to its parent component.	Creates a mutable ref object that persists across renders.
Primary Use Case	Customizing the interface between parent and child components.	Accessing and managing DOM nodes or other mutable values.
Syntax	`useImperativeHandle(ref, createHandle, [deps])`	`const ref = useRef(initialValue)`
Parameters	- `ref`: Ref object to attach custom methods or properties. - `createHandle`: Function that returns an object with methods or properties to expose. - `[deps]` (optional): Dependency array to specify when the handle should be recreated.	- `initialValue`: Initial value for the ref.
Return Value	- None	- Ref object with a `.current` property.

Example:

App.js

```
import React, { useRef } from "react";
import Child from "./Child";

function App() {
  const ref = useRef(null);
  return (
    <>
      <button onClick={() => ref.current.increment()}>Parent Click</button>
      <br />
      <Child ref={ref} />
    </>
  );
}

export default App;
```

Child.js

```
import React, { forwardRef, useImperativeHandle, useState } from "react";

const Child = forwardRef((props, ref) => {
  const [count, setCount] = useState(0);
  useImperativeHandle(ref, () => ({
    increment,
  }));

  function increment() {
    setCount(count + 1);
  }
  return (
    <>
      <button onClick={increment}>Child Click</button>
      <br />
      {count}
    </>
  );
});

export default Child;
```